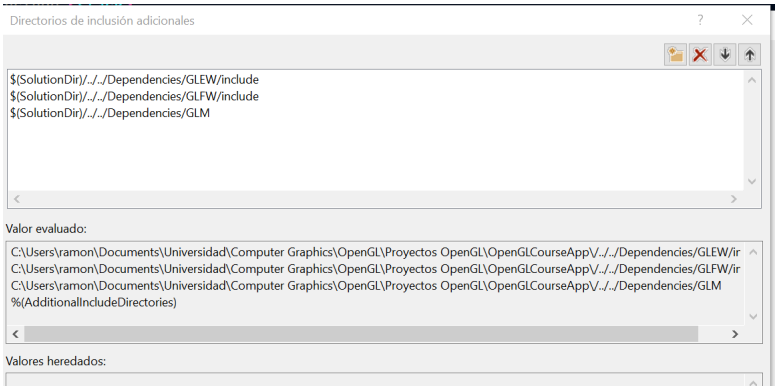
Proyecto: The Puppet
Computer Graphics

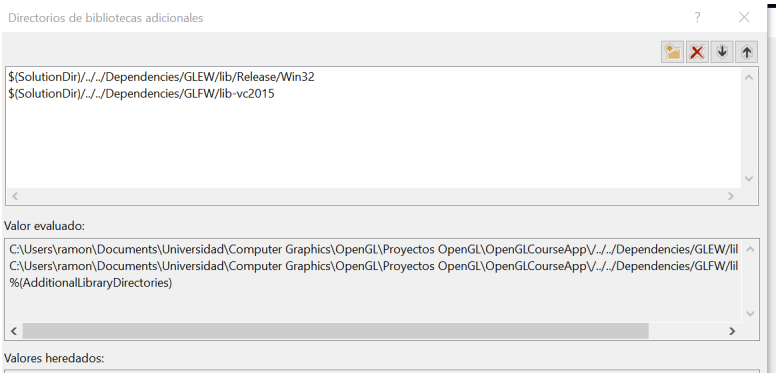
Luis Ramón Guajardo
luis.guajardo@cimat.mx
Universidad de Guanajuato
13 de octubre de 2022

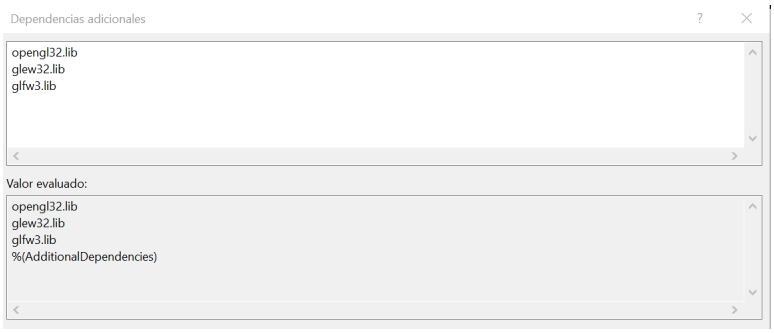
Distribución del Proyecto

Todos los archivos del proyecto vienen contenidos en una carpeta comprimida llamada *Proyecto 1 - Luis Ramon Guajardo*. Dentro de la carpeta, se encuentra otra llamada *OpenGLCourseApp*, creada como un proyecto de Visual Studio, donde dentro viene todo lo necesario para la compilación y ejecución del código. Dentro de la carpeta mencionada, hay otra con el mismo nombre y dentro de esta última hay una carpeta llamada *Shaders*, donde se pueden encontrar el Vertex y el Fragment shader en sus archivos correspondientes.

Cabe destacar que es de suma importancia que, a la hora de querer probar el código en una computadora distinta a la de origen, se haya realizado todo el proceso para establecer *OpenGL* como se vio en clase, es decir, que las librerias GLEW y GLFW estén instaladas y enlazadas a los directorios de inclusión y bibliotecas desde Visual Studio por ejemplo. Otra librería usada para este proyecto y necesaria para su funcionamiento es la de GLM, por lo que también debe estar enlazada de la misma forma que las dos anteriores. En el caso de mi computadora, los tengo enlazados desde Visual Studio como se muestra a continuación:







Nótese que donde tengo las librerias GLFW, GLEW y GLM es en una carpeta llamada *Dependencies*, la cual está adjunta en los archivos del proyecto como carpeta comprimida. Esta carpeta se puede poner dentro de la misma carpeta del proyecto, o bien, se puede dejar fuera, siempre y cuando se enlace correctamente. En mi caso, yo puse la carpeta fuera de la del proyecto para que pudiera servirme para otros proyectos aparte de este.

Instrucciones de Ejecución

Una vez que se tenga todo listo para probar el proyecto, se puede ejecutar desde Visual Studio con el Depurador local y aparecerá una ventana mostrando la marioneta en 3D. Se tiene implementada una camara que permite trasladarnos por el "mundo" de la marioneta y observarla de distintos ángulos. La cámara se controla con las siguientes teclas:

- **W**: Mover cámara hacia adelante
- **A**: Mover cámara hacia la izquierda
- **S**: Mover cámara hacia atrás
- **D**: Mover cámara hacia la derecha
- **Movimiento del mouse**: Mover el ángulo de la cámara

Por otra parte, se puede también controlar el movimiento de las distintas partes del cuerpo de la marioneta. Todos los ángulos de movimiento están restringidos para que el movimiento sea natural y se pueden mueven con las siguientes teclas numéricas:

- **1**: Mover el torso
- **2**: Mover la cabeza
- **3**: Mover el brazo superior izquierdo
- **4**: Mover el brazo inferior izquierdo
- **5**: Mover el brazo superior derecho
- **6**: Mover el brazo inferior derecho
- **7**: Mover el muslo izquierdo
- **8**: Mover la pantorrilla izquierda
- **9**: Mover el muslo derecho
- **0**: Mover la pantorrilla derecha

Por último, también se utilizan las teclas a continuación:

- **R**: Resetear a la configuración inicial de la marioneta
- **Esc**: Salir del programa

Problemas encontrados

Como tal, en el código y programa actual, considero que no hay ningun problema que afecte su funcionamiento y/o ejecución. No obstante, en el proceso de crear el proyecto sí me enfrenté a algunos problemas. El primero y considero que el más importante fue la falta de conocimiento y experiencia que tenía para poder implementar todo lo visto en clase a un código, por lo que, para solucionar esto, decidí junto con varios compañeros pagar un curso de OpenGL en la plataforma *Udemy*. Este curso me ayudó bastante a comprender y saber organizar bien toda la estructura del código, así como a ir creando desde cero todos los componentes programables del *Rendering Pipeline*, como los shaders, buffers y las matrices de modelado, proyección y vista.

Otro problema que tuve, fue que no sabía como implementar la jerarquía de movimientos para las partes de la marioneta, dado que el curso de *Udemy* antes mencionado no cubría ese tema como tal. La forma en que solucioné el problema fue a través de investigar en el capitulo 5 del Gortler y consultar distintos materiales de internet que explicaban como establecer la jerarquía con pilas de matrices de transformación.

Nota: La razón de que la carpeta de mi proyecto se llame OpenGLCourseApp y que la estructura de mi código (clases, nombres de funciones, etc.) sea probablemente similar a algunos de mis compañeros se debe a que varios estuvimos siguiendo el curso de Udemy, sin embargo, todo fuera de lo antes mencionado es completamente de mi autoría.

Referencias

- **Curso Udemy**: https://www.udemy.com/course/graphics-with-modern-opengl/
- **Material para jerarquía de movimientos**:
  - https://courses.cs.washington.edu/courses/csep557/03au/lectures/hierarchical-modeling.pdf
  - https://www.cs.utexas.edu/ theshark/courses/cs354/lectures/cs354-13.pdf
  - http://graphics.cs.cmu.edu/nsp/course/15-462/Spring04/slides/05-hierarchy.pdf
  - S.J. Gortler. Foundations of 3D Computer Graphics. MIT Press.